Motion Vector Recovery for Real-time H.264 Video Streams

Kavish Seth[1], Tummala Rajesh[1], V. Kamakoti[2], and S. Srinivasan[1]
[1] Dept. of Electrical Engg., Indian Institute of Technology Madras, Chennai, India
Email: {kavishseth, trajeshreddy}@gmail.com, srini@ee.iitm.ac.in
[2] Dept. of Computer Sc. and Engg., Indian Institute of Technology Madras, Chennai, India
Email: veezhi@gmail.com

***Abstract*— Among the various network protocols that can be used to stream the video data, RTP over UDP is the best to do with real time streaming in H.264 based video streams. Videos transmitted over a communication channel are highly prone to errors; it can become critical when UDP is used. In such cases real time error concealment becomes an important aspect. A subclass of the error concealment is the motion vector recovery which is used to conceal errors at the decoder side. Lagrange Interpolation is the fastest and a popular technique for the motion vector recovery. This paper proposes a new system architecture which enables the RTP-UDP based real time video streaming as well as the Lagrange interpolation based real time motion vector recovery in H.264 coded video streams. A completely open source H.264 video codec called FFmpeg is chosen to implement the proposed system. Proposed implementation was tested against the different standard benchmark video sequences and the quality of the recovered videos was measured at the decoder side using various quality measurement metrics. Experimental results show that the real time motion vector recovery does not introduce any noticeable difference or latency during display of the recovered video.**

***Index Terms*—Digital Video, Motion Vector, Error Concealment, H.264, UDP, RTP**

I. Introduction

Streaming of videos is a very common mode of video communication today. Its low cost, convenience and worldwide reach have made it a hugely popular mode of transmission. The videos can either be a pre-recorded video sequence or a live video stream. The videos captured are raw videos, which take up lot of storage space. Video compression technologies have to be widely employed in video communications systems in order to meet the channel bandwidth requirements.

The H.264 is currently one of the latest and most popular video coding standard [1]. Compared to previous coding standards, it is able to deliver higher video quality for a given compression ratio, and better compression ratio for the same video quality. Because of this, variations of H.264 are used in many applications including HD-DVD, Blu-ray, iPod video, HDTV broadcasts, and most recently in streaming media.

The compressed videos are sent in the form of packets for streaming media. The packets sent are highly prone to erroneous transmission. The packet may be damaged or may not be received at all. Such errors are likely to damage a *Group of Blocks* (GOB) of data in the decoded frames for block-based coding schemes such as H.264. Error also propagates due to high correlation between neighboring frames and degrades the quality of successive frames. *The Real Time Protocol* (RTP) over *User Datagram Protocol* (UDP) is the recommended and commonly used mechanism employed while streaming the H.264 media format [2].

Various approaches have been used to achieve error resilience in order to deal with the above problem. A nice overview of such methods is given in [3], [4]. One of the ways to overcome this problem is the implementation of *Error Concealment* (EC) at the decoder side. *Motion Vector Recovery* (MVR) is one way of EC which uses several mathematical techniques to recover the erroneous motion fields. Among the various MVR techniques reported in the literature, the *Lagrange Interpolation* (LAGI) is the fastest and a popular MVR technique which produces the high quality of the recovered video [5].

FFmpeg is a comprehensive multimedia encoding and decoding library that consists of numerous audio, video, and container formats [6]. This paper proposes a new system architecture which enables the real time video streaming as well as the real time MVR. The proposed architecture is implemented in both FFmpeg coder and decoder. A RTP packet encapsulation/decapsulation module is added to the FFMpeg codec, which packs/unpacks the *Network Abstraction Layer* (NAL) packets [2], [7] into single NAL unit type RTP packets. A UDP socket program is used at both coder and decoder sides to stream the RTP packets over UDP. A LAGI based MVR technique is implemented at decoder side. The proposed system implementation was tested against the different benchmark video sequences. Quality of the received videos can be measured using various quality measurement standards such as *Peak Signal to Noise Ratio* (PSNR) and *Video Quality Evaluation Metric* (VQM) [8] tools. The experimental section presents a brief analysis of which quality measurement parameter is best suited for the streaming video analysis. Experimental results show that the proposed implementation does not introduce any latency or degradation in the quality of the recovered video

DOI: 01.IJSIP.02.01.231

while streaming and performing the MVR simultaneously in real time.

Rest of the paper is organized as follows: a brief introduction of the H.264 codec is presented in Section II. An overview of real time streaming for H.264 videos is given in Section III. The MVR details are covered in Section IV. A system architecture for real time streaming and real time MVR is proposed in Section V. Experimental results are presented in Section VI and last section concludes the paper.

II. H.264 CODEC

The H.264 video codec is an efficient coding scheme that covers all forms of digital compressed video ranging from low bit-rate Internet streaming applications to HDTV broadcast and Digital Cinema applications. Compared to the current state of technology, the H.264 standard is shown to yield same quality of images as that produced by current state-of-the-art standards with a savings of 50% on the bitrate over the other standards. For example, the H.264 is reported to have achieved the same quality of images using a bitrate of 1.5 Mbit/s compared to what was achieved using the MPEG 2 standard at 3.5 Mbit/s [9].

The codec specification [1] itself distinguishes conceptually between a *video coding layer* (VCL) and a NAL. The VCL performs the signal processing part of the codec namely, mechanisms such as transform, quantization, and motion compensated prediction; and a loop filter. It follows the general concept of most of today's video codecs, a *Macro Block* (MB) based coder that uses inter picture prediction with motion compensation and transform coding of the residual signal. The VCL encoder outputs slices: a bit string that contains the MB data of an integer number of MBs, and the information of the slice header (containing the spatial address of the first MB in the slice, the initial quantization parameter, and similar information). The NAL encoder encapsulates the slice output of the VCL encoder into NAL units, which are suitable for transmission over packet networks or use in packet oriented multiplex environments.

A NAL unit consists of a one-byte header and the payload byte string. The header indicates the type of the NAL unit, presence of bit errors or syntax violations in the NAL unit payload, and information regarding the relative importance of the NAL unit for the decoding process.

III. REAL TIME STREAMING FOR H.264 VIDEOS

The *Internet Protocol* (IP) [10] is a packet-based-network transport protocol upon which the internet is built. IP packets are encapsulated in lower, hardware-level protocols for delivery over various networks (Ethernet, etc), and they encapsulate higher transport- and application-level protocols for streaming and other applications. In the case of streaming H.264 videos over IP networks, multiple protocols, such as RTP and UDP, are carried in the IP payload, each with its own header and payload that recursively carries another protocol packet. For example, H.264 video data that is carried in an RTP packet which in turn is carried in a UDP packet which in turn is carried in an IP packet. The UDP sends

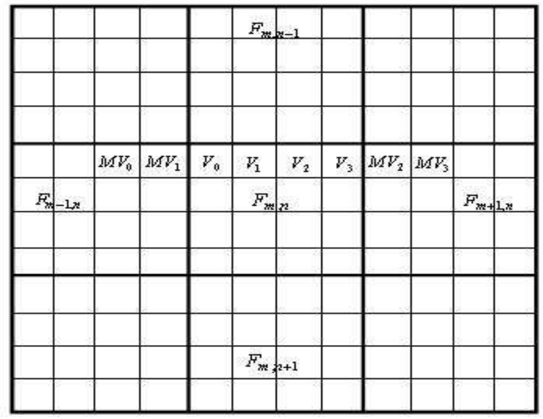

Figure 1. Frame with Lost Macro block

the media stream as a series of small packets. This is simple and efficient; however, there is no mechanism within the protocol to guarantee delivery. It is up to the receiving application to detect loss or corruption and recover data using error correction techniques. If data is lost, the stream may suffer a dropout.

The RTP was developed for carriage of real time data over IP networks [11], [12]. RTP is a native internet protocol, designed for and fitting well in the general suite of IP protocols. RTP does not provide any multiplexing capability. Rather, each media stream is carried in a separate RTP stream and relies on underlying encapsulation, typically UDP, to provide multiplexing over an IP network. Because of this, there is no need for an explicit de-multiplexer on the client either. Each RTP stream must carry timing information that is used at the client side to synchronize streams when necessary.

IV. MOTION VECTOR RECOVERY IN H.264

Among the existing MVR algorithms reported in literature, the LAGI [5] is the most popular techniques used to recover the lost MVs in H.264 encoded video. The computation cost of the LAGI based MVR technique is lower than most other interpolation functions and hence this technique becomes a good choice for real time video streaming applications. This section presents a MVR method that is based on LAGI formula. For $n+1$ given points (x_i, y_i) and $i = 0,\dots,n$ by suitable choice of parameters, we can constitute the interpolation function $f(x)$ such that $f(x_i) = y_i$. The formula for LAGI function is as follows:

$$f(x) = y_0 L_0^{(n)}(x) + y_1 L_i^{(n)}(x) + \cdots + y_n L_n^{(n)}(x) \qquad (1)$$

DOI: 01.IJSIP.02.01.231

Where $L_0^{(n)},\ldots,L_n^{(n)}$ denote the parameters in the LAGI formula. They can be computed from the $n+1$ given points, by the following formula:

Table I
The Corresponding Coordinates Of Each MV

x	x_0	x_1	x_2	x_3
$y=f(x)$	MV_0	MV_1	MV_2	MV_3

$$L_i^{(n)}=\frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \tag{2}$$

The Lagrangian basis functions are calculated as follows:

Where,

$$L_0^{(3)}=\left[\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}\right] \tag{3}$$

$$L_1^{(3)}=\left[\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}\right] \tag{4}$$

$$L_2^{(3)}=\left[\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}\right] \tag{5}$$

$$L_3^{(3)}=\left[\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}\right] \tag{6}$$

(1e)

The Lagrangian polynomial is formed as follows:

$$f(x)=MV_0L_0^{(3)}(x)+MV_1L_1^{(3)}(x)+\cdots+MV_3L_3^{(3)}(x) \tag{7}$$

The H.264 standard divides every frame into several *Macro Blocks* (MBs). Each MB is associated with 1 to 16 *Motion Vectors* (MVs) ensuring backward compatibility with previous standards. Fig. 1 shows a H.264 frame segment with 9 MBs denoted by $F_{m,n}$, where m and n denote the spatial location of the MB within the frame. Each MB is associated with 16 MVs. In Fig. 1, let $F_{m,n}$ denote the lost MB. As in the case of many MVR algorithms, it is assumed that either two of the vertically adjacent or two of the horizontally adjacent MBs of the lost MB are correctly decoded [5], [13]. In Fig. 1, it is assumed without loss of generality that both the horizontally adjacent MBs of $F_{m,n}$ are error-free. In this case, the lost MVs of $F_{m,n}$ are recovered row-by-row. Let $MV_i\,(0\le i\le 3)$ denote the correct MVs that belong to a particular row of the horizontally adjacent MBs of $F_{m,n}$ as shown in Fig. 1. Let $V_i\,(0\le i\le 3)$ represent the MVs of the rows of $F_{m,n}$ that need to be recovered.

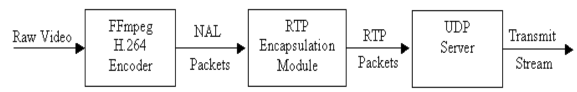

Figure 2. Proposed encoder architecture

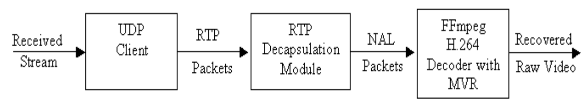

Figure 3. Proposed decoder architecture

The procedure to recover the one row of MVs of $F_{m,n}$ is described as follows:

1. The correct neighboring MVs $MV_0,\ldots,MV_3$ are used to compute Lagrange basis functions by substituting these values in (3) – (6).
2. By substituting in (7), the four (x,y) pairs shown in Table I, a third degree LAGI polynomial is formed.
3. The lost MVs are computed by substituting the values of MV_i and x_i in (7).

A similar procedure is followed if the vertically adjacent frames of $F_{m,n}$ are error-free. In this case, the lost MVs of $F_{m,n}$ are recovered column-by-column using the correct MVs of $F_{m,n-1}$ and $F_{m,n+1}$ as shown in Fig. 1.

V. Proposed Architecture

FFmpeg provides the complete package to encode/decode most of the popular encoded videos. It is a computer program that can record, convert and stream digital audio and video in numerous formats [6]. FFmpeg is a command line tool that is composed of a collection of free software and open source libraries. It includes libavcodec, an audio/video codec library used by several other projects, and libavformat, an audio/video container mux and demux library. On careful examination of the libavcodec source code, the file used by FFmpeg to decode the H.264 videos was found to be ”h264.c”.The name of the project comes from the MPEG video standards group, together with ”FF” for ”fast forward”. FFmpeg is developed under Linux, but it can be compiled under most operating systems, including Apple Inc. Mac OS X, Microsoft Windows and AmigaOS. The proposed system architecture at FFmpeg encoder side consists of a H.264 encoder, a RTP encapsulation module and a UDP server. This UDP server uses a socket program to transmit any

DOI: 01.IJSIP.02.01.231

H.264 video as the UDP packets over a predetermined port known both to the server as well as to the client.

Table Ii
Table Showing The Psnr Values For Various Video Sequences

Sequence	PSNR (dB)		
	without error	with MVR	with error
Akiyo	41.32	36.34	16.26
Coastguard	37.79	32.78	8.34
Foreman	38.80	29.20	9.40

Table Iii
Table Showing The Vqm Values For Various Video Sequences

Sequence	VQM		
	without error	with MVR	with error
Akiyo	0.68	1.10	7.8
Coastguard	1.00	1.76	19.8
Foreman	0.83	2.14	15.56

The block diagram of FFmpeg encoder side architecture is presented in Fig. 2. The system architecture at FFmpeg decoder side consists of a H.264 decoder, a RTP decapsulation module and a UDP client which keeps listening on same port used by socket program run on server end. The block diagram of FFmpeg decoder side architecture is presented in Fig. 3. Once UDP client receives the H.264 live stream, it forwards its payload (UDP payload is a single NAL unit type RTP packets) to the RTP decapsulation module. Note that the RTP encapsulation/decapsulation modules can deal only with single NAL unit type of packets. RTP decapsulation module gives out the NAL packets to the H.264 decoder with MVR capability. The RTP decapsulation module also performs a check on erroneous or missing packets based on the RTP sequence number and timestamp and it reports the check results to the decoder to perform MVR. In order to introduce MVR as a part of decoding process, a LAGI based MVR module is incorporated with the motion compensation module of the FFmpeg decoder.

VI. Experimental Results

After implementing the proposed system architecture in FFmpeg codec, the implementation is tested with three different standard benchmark video sequences the Akiyo, the Foreman, and the Coastguard. All of them are *Quarter Common Intermediate Format* (QCIF) raw sequences and the total number of frames in each sequence is 70. In order to conduct the experiments, errors are deliberately introduced into these sequences before streaming them to decoder. Note that 15% MB loss randomly is introduced in all the P frames of each sequence and the *Quantization Parameter* (QP) is set to 24.

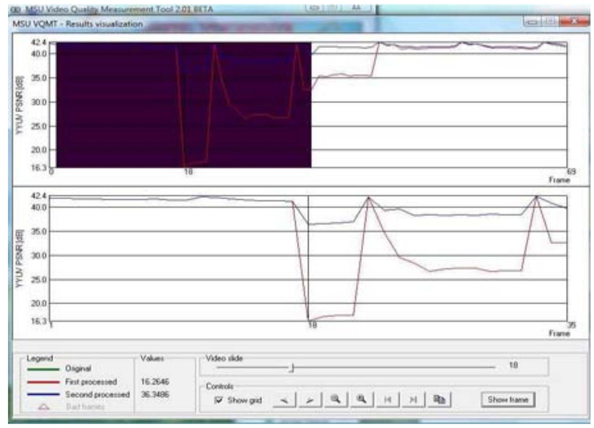

Figure 4. PSNR of the Akiyo sequence with error and after MVR

Figure 5. The frame in the Akiyo sequence without error

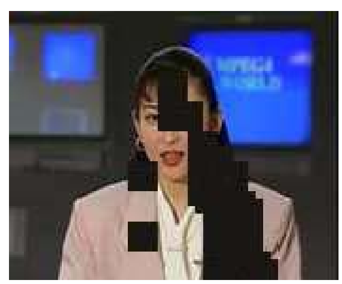

Figure 6. The error frame in the Akiyo sequence

The complexity and cost of subjective quality measurement make it attractive to be able to measure quality automatically using an algorithm. The most widely used measure is *Peak Signal to Noise Ratio* (PSNR) but the limitations of this metric have led to many efforts to develop more sophisticated measures like *Video Quality Evaluation Metric* (VQM) that approximate the response of real human observers. VQM is a modified discrete cosine transform based VQM based on Watsons proposal [14], which exploits the property of visual perception. A detailed account of how it is implemented can be found at [8]. The

DOI: 01.IJSIP.02.01.231

values of PSNR and VQM have been calculated using a Video Quality Measurement tool called MSU [8].

Figure 7. The error frame in the Akiyo sequence after MVR

The PSNR values for the above mentioned video sequences are given in Table II. The PSNR graph for the Akiyo sequence with error and after MVR is shown in Fig. 4. The frames of the Akiyo sequence without error, with error and after MVR are shown in Fig. 5, 6 and 7 respectively.

The VQM values are used to quantify the quality of the video streams. The VQM values reflects the different characteristics of a video stream e.g. latency, frames drops, visual quality of a frame, resolution etc. Higher the value of VQM of a given video stream indicates the bad quality. On the other hand, PSNR values are used to access only the visual quality of frames and doesn't consider other parameters like latency, frame drops etc. The VQM values for the different video sequences are given in Table III. It is clear from the Table III that the VQM values of the recovered videos fall within pretty much acceptable range. This indicates that the streaming quality on the proposed implementation is quite good. The VQM graph for the Akiyo sequence with error and after MVR is shown in Fig. 8. Though PSNR and VQM can both be used to measure video quality, VQM performances much better in situations when PSNR fails. The light computation and memory load make it even more attractive for wide applications [8].

CONCLUSIONS

Video streaming using the RTP over the UDP has been successfully tested and reported in this paper. The MVR using the Lagrangian interpolation technique has been integrated into FFmpeg and tested for various erroneous H.264 video sequences. The MVR algorithm also gives good results with real time video without any noticeable difference or latency during display. Also the analysis of the received videos through different quality measurement metrics. shows that the VQM is a better way to measure quality than PSNR.

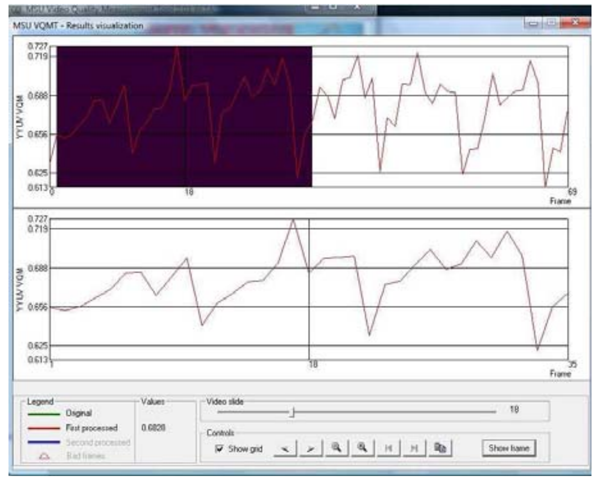

Figure 8. VQM of the Akiyo sequence with error and after MVR

REFERENCES

[1] ITU-T Recommendation: *H.264 Advanced video coding for generic audiovisual services*, May 2003.
[2] S. Wenger, M. M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer, *RTP Payload Format for H.264 Video*: RFC 3984, February 2005.
[3] Y. Wang and Q. F. Zhu, "Error control and concealment for video communication: a review," *Proc. IEEE*, vol. 86, no. 5, May 1998
[4] Y. K. Wang, M. M. Hannuksela, V. Varsa, A. Hourunranta, and M. Gabbouj, "The error concealment feature in the H.26L test model," *Proc. IEEE Int. Conf. Image Processing*,,pp.729-732, 2002.
[5] J. H. Zheng and L. P. Chau, "Motion vector recovery algorithm for digital video using Lagrange Interpolation," *IEEE Trans. Broadcasting*, vol. 49, No. 4, pp. 383-389, Dec 2003.
[6] FFmpeg, http://ffmpeg.mplayerhq.hu/.
[7] Iain E. G. Richardson, *H.264 and MPEG-4 Video Compression*: Wiley, 2003.
[8] MSU, *Video Quality Measurement Tools*: VQM Reference Manual, http://www.compression.ru/video.
[9] A. Luthra, G. J. Sullivan, and T. Wiegand, "Special Issue on H.264/AVC," *IEEE Transactions Circuits and Systems on Video Technology*, vol. 13, no. 7, pp. 560-576, July 2003.
[10] J. Postel, *Internet Protocol*: RFC 791, September 1981.
[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*: STD 64, RFC 3550, July 2003.
[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*: RFC 1889, January 1996.
[13] Kavish Seth, M. Komisetty, V. Anand, V. Kamakoti, and S. Srinivasan, "VLSI implementation of Motion Vector Recovery Algorithms for H.264 Video Codecs," *13th IEEE/VSI VLSI Design and Test Symposium*, Bangalore (India), July 2009.
[14] A. B. Watson, *Image data compression having minimum perceptual error*: US Patent 5,629,780. 1997.

DOI: 01.IJSIP.02.01.231